Title: ADVANCED SPAM DETECTION TECHNIQUES

TECHNICAL FIELD

This invention is related to systems and methods for identifying spam messages, and more particularly that find characteristics that are difficult for spammers to avoid and characteristics in non-spam that are difficult for spammers to duplicate.

BACKGROUND OF THE INVENTION

The advent of global communications networks such as the Internet has presented commercial opportunities for reaching vast numbers of potential customers. Electronic messaging, and particularly electronic mail ("email"), is becoming increasingly pervasive as a means for disseminating unwanted advertisements and promotions (also denoted as "spam") to network users.

The Radicati Group, Inc., a consulting and market research firm, estimates that as of August 2002, two billion junk e-mail messages are sent each day - this number is expected to triple every two years. Individuals and entities (*e.g.*, businesses, government agencies) are becoming increasingly inconvenienced and oftentimes offended by junk messages. As such, spam is now or soon will become a major threat to trustworthy computing.

Common techniques utilized to thwart spam involve the employment of filtering systems/methodologies. One proven filtering technique is based upon a machine learning approach. Machine learning filters assign to an incoming message a probability that the message is spam. In this approach, features typically are extracted from two classes of example messages (*e.g.*, spam and non-spam messages), and a learning filter is applied to discriminate probabilistically between the two classes. Since many message features are related to content (*e.g.*, whole words and phrases in the subject and/or body of the message), such types of filters are commonly referred to as "content-based filters". These types of machine learning filters usually employ exact match techniques in order to detect and distinguish spam messages from good messages.

Unfortunately, often spammers can fool conventional machine learning and/or content-based filters by modifying their spam messages to look like good mail or to include a variety of erroneous characters throughout the message to avoid and/or confuse character recognition systems. Thus, such conventional filters provide limited protection against spam.

SUMMARY OF THE INVENTION

The following presents a simplified summary of the invention in order to provide a basic understanding of some aspects of the invention. This summary is not an extensive overview of the invention. It is not intended to identify key/critical elements of the invention or to delineate the scope of the invention. Its sole purpose is to present some concepts of the invention in a simplified form as a prelude to the more detailed description that is presented later.

Spam filters, whether based on machine-learning or on other techniques, must look at the contents of a message to determine whether a message is spam or not. Unfortunately, spammers are sometimes able to disguise many aspects of their messages. They are able to misspell spam-like words, use synonyms, or use images that include words. While a spam filter could use optical character recognition (OCR) software to find the words in images, this is typically too expensive, especially if spammers intentionally use images that are difficult for OCR systems. To mitigate the ability to disguise their messages, features can be generated which are difficult for spammers to fake.

Features are facts detected by an email or message parsing component. The message parsing component can create a feature for each word in the message. It can also create a feature each time punctuation is used which can be dependent upon the kind of punctuation used. Features can be used either by machine learning filters or in many other ways, such as a part of hand-built rules.

The subject invention provides for a system and method that facilitates detecting and preventing spam by including additional features beyond those typically used by conventional spam filters which are difficult for spammers to fake. One such feature involves looking at the pairs of features in a message. Certain features in spam are easily

forged, or of little value, when considered separately, but are much more valuable together – that is, when they are considered together. Exemplary features which can be employed as pairs include those derived from or related to the origination information of a message. In particular, a domain and host name in a SMTP (Simple Mail Transfer Protocol), the domain and host name in a HELO command, an IP address or subnet in a Received from header, any domain or host name in a display name, any domain or host name in a Message From field, and any time zones in the last received from header should all match in some way or combination. Hence, pairs of any of the above can be useful for training a machine learning filter or any other rule-based filter.

A second feature involves examining a run of characters. Most conventional features in messages are related to words in the messages, and most typically to space-separated words. However, the fact that a certain character sequence (with or without spaces) occurs in a part of a message can be indicative of spam. Thus, the present invention provides for a system and method that employ features created for each sequence of characters or substantially all possible sequences of characters, including punctuation and spaces. Some spammers may also include chaff at the ends or beginnings of subject lines or messages which can disrupt exact match techniques found in most spam filtering systems. The chaff can include character n-grams such as "xz" or "qp" that rarely occurs in good mail. Thus, the presence or occurrence of chaff and/or character n-grams can be strong indicators that the message is bad (*e.g.*, spam). The character n-gram can also be position-dependent. Accordingly, features including this position dependence can also be created and employed in accordance with the subject invention.

An alternative to using rare character sequences to detect chaff involves yet a third type of feature that can be employed in machine learning systems. The third feature involves detecting high entropy of characters using a character n-gram language model, for example. In this model, a probability of occurrence can be assigned to each character such that certain character sequences are more probable to occur than others. For instance, the character sequence "he" (*e.g.*, as found in "the", "hear", "she", "theater", etc.) is more likely to occur than the sequence "xz" in any given run or string of

characters. Thus, the entropy for the character sequence "xz" will be higher than it will be for the sequence "he".

In addition to high entropy, an average entropy of characters can also be detected such as at the end or the beginning of a subject line or of a message. Furthermore, features relating to the relative entropy of characters can be useful. For instance, features can be designated for when an average entropy at the beginning of a subject line is 0.5 higher than the average entropy at the middle of the subject line. Other exemplary features could correspond to an average entropy at the end of a message body being 1.0 more than at the middle of the message. Moreover, each of these detected events of high, average, and/or relative entropy can be employed as separate features.

A fourth type of useful features involves generic headers. Traditional machine learning algorithms only use common features in the subject line and body of messages or features based on other common fields found in a message header. Unlike traditional filters, the present invention utilizes substantially all headers, including the presence or absence of header line types. More importantly, the present machine learning systems can automatically identify all useful header features and in some cases, can even exclude some header lines as well.

According to other aspects of the present invention, additional features of electronic mail (email) communications which can be useful to machine learning techniques include extended sizes of features as well as image features. Since very little spam is very big, the many different sizes of features in combination with at least one other feature discussed hereinabove can facilitate identification of spam. For instance, features can be created to correspond to message size. That is, for message sizes greater than 100 bytes, 200 bytes, and up to ***b*** bytes (wherein ***b*** is greater than or equal to 1), a feature can be generated for each size or size range. This can also be applied to subject line and display name sizes since spammers often use lengthy display names to confuse and/or disguise the source of the message. Similarly, subject lines of spam tend to include a significant portion of or the entire body of the message since some users never open their messages but instead rely on the subject line alone.

Any of the above described features can be used by machine learning systems to train and improve junk mail and/or spam filters, thereby making it more difficult for

spammers to modify their messages around these filters. Moreover, spammers are left with fewer opportunities for spammers to get their spam through messaging systems.

To the accomplishment of the foregoing and related ends, certain illustrative aspects of the invention are described herein in connection with the following description and the annexed drawings. These aspects are indicative, however, of but a few of the various ways in which the principles of the invention may be employed and the present invention is intended to include all such aspects and their equivalents. Other advantages and novel features of the invention may become apparent from the following detailed description of the invention when considered in conjunction with the drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a general block diagram of a system that facilitates preventing spam in accordance with an aspect of the present invention.

Fig. 2 is a schematic diagram of a break down of features from a HELO command in accordance with an aspect of the present invention.

Fig. 3 is a schematic diagram of a break down of features from a MAIL FROM command in accordance with an aspect of the present invention.

Fig. 4 is a schematic diagram of a break down of features from a DATA command in accordance with an aspect of the present invention.

Fig. 5 is a schematic diagram of a break down of features that can be paired up from a Message From line and from a Received line in accordance with an aspect of the present invention.

Fig. 6 is a general block diagram of a system that facilitates generating features relating to runs of characters and/or character sequences and/or entropy of such character sequences in accordance with an aspect of the present invention.

Fig. 7 is a general block diagram of a system that facilitates generating features relating to message header content and/or size-related features and/or images present in the message in accordance with an aspect of the present invention.

Fig. 8 is a flow diagram of an exemplary method that facilitates creating features including pairs of features to train a filter in accordance with an aspect of the present invention.

Fig. 9 is a flow diagram of an exemplary method that facilitates employing the trained filter of Fig. 8 to identify spam and/or spam-like messages.

Fig. 10 is a flow diagram of an exemplary method that facilitates creating features based on a run of characters and/or on entropy of such run of characters which can be used to train a filter in accordance with an aspect of the present invention.

Fig. 11 is a flow diagram of an exemplary method that facilitates employing the trained filter of Fig. 10 to identify spam and/or spam-like messages.

Fig. 12 is a flow diagram of an exemplary method that facilitates creating features which can be used to train a filter in accordance with an aspect of the present invention.

Fig. 13 is a flow diagram of an exemplary method that facilitates employing the trained filter of Fig. 12 to identify spam and/or spam-like messages.

Fig. 14 is a schematic block diagram of an exemplary communication environment in accordance with the present invention.

DETAILED DESCRIPTION OF THE INVENTION

The present invention is now described with reference to the drawings, wherein like reference numerals are used to refer to like elements throughout. In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It may be evident, however, that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to facilitate describing the present invention.

As used in this application, the terms "component" and "system" are intended to refer to a computer-related entity, either hardware, a combination of hardware and software, software, or software in execution. For example, a component may be, but is not limited to being, a process running on a processor, a processor, an object, an executable, a thread of execution, a program, and/or a computer. By way of illustration, both an application running on a server and the server can be a component. One or more components may reside within a process and/or thread of execution and a component may be localized on one computer and/or distributed between two or more computers.

The subject invention can incorporate various inference schemes and/or techniques in connection with generating training data for machine learned spam filtering. As used herein, the term "inference" refers generally to the process of reasoning about or inferring states of the system, environment, and/or user from a set of observations as captured *via* events and/or data. Inference can be employed to identify a specific context or action, or can generate a probability distribution over states, for example. The inference can be probabilistic – that is, the computation of a probability distribution over states of interest based on a consideration of data and events. Inference can also refer to techniques employed for composing higher-level events from a set of events and/or data. Such inference results in the construction of new events or actions from a set of observed events and/or stored event data, whether or not the events are correlated in close temporal proximity, and whether the events and data come from one or several event and data sources.

It is to be appreciated that although the term message is employed extensively throughout the specification, such term is not limited to electronic mail *per se*, but can be suitably adapted to include electronic messaging of any form that can be distributed over any suitable communication architecture. For example, conferencing applications that facilitate a conference between two or more people (*e.g.*, interactive chat programs, and instant messaging programs) can also utilize the filtering benefits disclosed herein, since unwanted text can be electronically interspersed into normal chat messages as users exchange messages and/or inserted as a lead-off message, a closing message, or all of the above. In this particular application, a filter could be trained to automatically filter particular message content (text and images) in order to capture and tag as junk the undesirable content (*e.g.*, commercials, promotions, or advertisements). To give another example, SMS messages for cellular phones can also be filtered.

Referring now to Fig. 1, there is illustrated a general block diagram of a system 100 that facilitates using pairs of features to promote spam detection in accordance with an aspect of the present invention. Some features are especially useful as a pair, even when they are already employed individually. In general, there is information where mail is from that should match. For example, one's IP address is not practical to forge. Thus, a spam filter could be trained to check that some other features are consistent with the IP

address feature. Match can be used in a general sense. For instance, the same servers are used to send mail from two domains (*e.g.*, Hotmail and MSN); the HELO command and the "from" command need not provide the same domain, but generally will only occur in certain valid pairs.

As shown in Fig. 1, messages from one or more senders 110 are delivered from their respective sender(s) to a mail server 130, such as an SMTP server, included in the system 100. Delivery of the messages can be accomplished through a number of SMTP commands 120, for example. Other mail delivery protocols are possible and can be applied to the present invention in a similar manner.

Accordingly, a series of SMTP commands can be issued and resolved before the message 140 is accepted by the receiver for delivery. In particular, origination information 150 that can be used for establishing the pairs of features can be found in the SMTP commands 120. In order to derive features from the origination information 150, the information 150 can be evaluated and parsed by a message parsing component 160. After at least a portion of the origination information 150 of the message 140 has been parsed, the parsed portions or features can be communicated to a feature pairing component 170. The feature pairing component 170 can analyze any possible combination of features such that a resulting pair of features is useful as an additional feature. A filter training component 180, which is operatively coupled to the system 100, can make use of the feature pairs when training a spam filter, for example.

Once the filter is sufficiently trained, it can be employed in conjunction with a machine learning system and applied to another group of mail messages to filter spam-like messages from the group. The filter can be periodically updated and/or new filters can be made as needed to effectively distinguish legitimate mail from spam mail.

Figs. 2-4 depict various features that can be parsed from the origination information found within SMTP commands and that can be combined into a number of useful pairs (*e.g.*, as indicated by the arrows between the figures) in accordance with an aspect of the present invention. For example, in Fig. 2, the first SMTP command can be a HELO command 200 in which the sending machine says its name, such as for instance, HELO *x.y.z*. If *x.y.z* is of the form mail1.ddd.com, then "mail1.ddd.com" can be referred

to as the host name 210 and "ddd.com" can be referred to as the domain name 220. Hence, host names 210 can be stripped down to domain names 220.

We can also detect the sender's IP address; the SMTP protocol typically is used over TCP/IP, and thus the IP address used for the communication is known to the receiver. IP addresses 230 are often sold or used in groups called subnets 240. Subnets 240 can be defined in various ways, though in practice, one exemplary subnet can be defined as including all IP addresses that share the first 24 bits. Therefore, if the HELO command 200 says HELO ddd.com, there may be multiple machines sending from ddd.com; however, most of the sending machines will be on the same subnet 240.

In general, some pairs of features make less sense than others. For example, the pairing of HELO host name 210 and HELO domain name 220 is less useful since one is derived from the other. However, the pairing of the subnet 240 and the HELO domain name 220 is very useful because at least a portion of these features should match under normal circumstances.

After the HELO command 200, a line comprising *x.y.z* and the IP address that this message is coming from and the time including an alleged time zone can be appended to a Received from line by a recipient of the message. A spam filter can scan the headers to see what the HELO command 200 said. The sender's alleged time zone (one feature) should match the time stamp (another feature) of the message. Furthermore, the alleged time zone should also match the sender's alleged machine name or IP address in the HELO command 200. A mismatch thereof can be indicative of spam.

Generally, the IP address from the HELO command 200 (*e.g.*, HELO x.y.z) should be the same or match the alleged machine name or IP address in the Received from line, but spammers can forge mail by not using the correct host or domain name for their IP address. Thus, a mismatch here can be indicative of spam. It should be appreciated that it is even less likely for the subnet 240 of the IP address 230 to mismatch a domain name 220 than it is for the IP address 230 to mismatch (a domain name 220).

In the case where some domains have not configured their machines correctly to provide the proper machine name at the HELO command 200, the filter can learn what the particular pair is such that when it sees the pair again (in subsequent messages), it can accept that pair as a proper match. Therefore, the filter can be trained to accommodate

personal preferences as well as minor errors or mismatches in the origination information as long as some consistency is maintained between training the filter and using the filter. In addition, a list of possible pairs populated with valid host names and valid IP addresses, for example, can be created such that anything detected in the origination information that is outside of this list is more likely to be spam.

A later command is a MAIL FROM command 300 as demonstrated in Fig. 3 in accordance with an aspect of the present invention. Also known as the Envelope From, the MAIL FROM command can be in the form MAIL FROM a@b.c.d. It should be understood that b.c.d may or may not be the same as x.y.z, though according to the example, it should be. In particular, parts of the host names 310 should typically match. For instance, c.d should be the same as or correspond to y.z to constitute a valid match by the filter. The host name 310 can be further stripped down to a domain name 320 to provide additional pairs of features such as with the HELO IP address 230 (Fig. 2).

Later in the message, such as during a DATA command 400, a line of the form From: e@f.g.h can be added. Again, host name 410 f.g.h can be the same as x.y.z and b.c.d. Alternatively, at least domain name 420 g.h should match y.z and c.d, but not always. The From line is also referred to as the Message from. Sometimes, the line will be of the form From: "i" <e@f.g.h>. The "i" is called a display name 430. Many email clients actually display only the display name "i" rather than e@f.g.h. However, "i" could be in the form of "j@k.l.m", thereby misleading the user about the identity of the message sender. This alone should be indicative of spam because such a display name is atypical. However, if "i" is present in the form "j@k.l.m", then k.l.m should match the other host names; or at the very least, the domain names should match (*e.g.*, l.m corresponding to g.h).

In some cases, it can be inconvenient or difficult to tell if a particular triple (*e.g.*, x.y.z) is a host name or a domain name. At times, a guess that it could be either is necessary. For instance, if the HELO command gives an address of the form x.y.z and the MAIL FROM command has an address of the form y.z, then it can be ascertained with some certainty that x.y.z is a host name and y.z is a domain name. If the HELO command gives an address of the form x.y.z and the MAIL FROM command gives an address of the form b.c.d, then a guess would have to be made that x.y.z and b.c.d are

both host names and domain names and that y.z and c.d are domain names. All pairs of guesses can be employed as features rather than just the best guess. Alternatively, the best guess can be used. A variety of simple heuristics for making these guesses can be deduced. In general, when dealing with machine learning systems, it is not important that the guessing always be correct as long as the guessing is consistent for a given kind of mail – so the same feature pairs occur consistently.

Moreover, features relating to the domain and host names in the SMTP MAIL FROM command, the domain and host names in the HELO command, the IP address or subnet in the Received from header, any domain or host name in the Display name, any domain or host name in the Message From, any time zones in the last Received from header, and a type of mailing software used by the sender should all match in some way. Pairs of any of these features are likely to be useful since substantially all of the listed attributes can be forged by spammers, with the exception of IP address and the subnet thereof. Hence, pairs that include the IP address or subnet are especially powerful and useful when combined with any of the other features.

Fig. 5 demonstrates possible pairs of features that can be derived from a Message From line and a Received from header (collectively 500) of a mail message. As shown, the domain name "domain.com" 510 can be paired to match an IP address 520, a sender's alleged time zone 530, and/or a subnet 540 of the IP address 520. Alternatively or in addition, the sender's alleged time zone 530 could be paired to match the sender's alleged IP address 520. Other features not illustrated herein as well as other pairs of features not demonstrated in the figure are possible.

Figs. 6 and 7 represent additional feature generation systems that facilitate advanced spam detection. Referring now to Fig. 6, there is illustrated a block diagram of an exemplary system 600 for creating features as they relate to a run of characters and features based at least in part on the entropy of these character sequences.

Most features in messages are related to words found in the messages. However, the presence of a certain character sequence in a message or in part of a message can also be useful. For instance, sometimes spammers use a character sequence such as "R.I.C.H." instead of "rich" or "RICH". Using a pattern-match technique, words written as "R.I.C.H." can be readily extracted to reveal the base word "RICH".

In addition, spammers sometimes add random letters as chaff to the ends or beginnings of subject lines or messages. This disrupts exact match techniques commonly employed in conventional filters. Since these random character sequences are likely to include character n-grams like "xz" or "qp" that rarely, if ever, occur in good mail, their occurrence in a message can be strong indicators that the message is bad (*e.g.*, spam). Spammers can also evade traditional spam filters by arbitrarily adding in punctuation such as periods and hyphens, as well as symbols, to distort words and/or phrases that are known to be characteristic of spam.

To mitigate this type of intentional manipulation, the system 600 generates features for each possible sequence of characters in order to identify and detect intentional character substitutions, insertions, and misspellings. The system 600 accomplishes this in part by walking through text, character by character, and generating features for each run of length ***n*** (*e.g.*, ***n*** is an integer greater than or equal to one), which will effectively pick up words, punctuation, spaces, and other content.

For example, a sender 610 sends a message 620 as shown in the figure. The message 620 is delivered to a message server 630 where it can be processed to yield one or more features by a character sequencing component 640. The character sequencing component 640 can analyze at least a portion of the message via searching for particular character sequences, strings and/or sub-strings that are indicative of spam. The sequences, strings and/or sub-strings are not necessarily whole or space-separated words.

For instance, imagine that the message 620 includes the text:

"Get Rich ~-quick-~ by Calling now!!!!!"

A run of length 6 would create these exemplary character sequences 650:

"Get Ric"

"et Rich"

"t Rich "

" Rich ~-" …

A run of length 7 would create these exemplary character sequences 650:

"Rich ~-q"

"ich ~-qu"

"now!!!!"

As the character sequences 650 are being identified and created, a feature generating component 660 generates corresponding features 670 for each character sequence. Such features 670 can then be used by a filter training component 680 to train a spam filter, for example.

Multiple run lengths from as few as one and up to some length ***n***, for example, for the same message can be utilized to keep track of both the individual lengths (strings) as well as sublengths (substrings).

With respect to character n-grams, the same or different feature can be used depending on where the n-gram occurs. N-grams may be located in From addresses, subject lines, text bodies, html bodies and/or attachments. Furthermore, n-gram features can be generated and employed according to their positions in the message. For instance, since chaff (*e.g.*, comprising n-grams) tends to occur at the beginning or end of a subject, a rare character sequence at the beginning or end of a subject line is more indicative of spam than a rare character sequence in the middle. Hence, the system 600 can be programmed to detect chaff and/or n-grams only at the desired positions, such as the beginning and end of the subject line. Similarly, n-gram features can also be position dependent for the beginning or the end of the message.

The system 600 is also valuable for use with foreign languages, especially those that do not separate words with spaces such as Korean and Japanese dialects (*e.g.*, Hiragana and Katakana). As described above, substantially all sequences of different character lengths can be readily detected. Alternatively, the system 600 can be invoked only when it is suspected that the text is in a foreign language, such as when there are few spaces, when many characters that are rarely used in English are detected (*i.e.*, high byte characters), or when certain Unicode character types are detected. Thus, character n-grams would only be used for characters determined to not have uppercase, lowercase, punctuation or space characteristics. For example, when the message is scanned and very few spaces and/or long strings of high byte characters are detected, then the n-gram sequencing can be invoked. This restricted application of character n-grams can be advantageous over using full character n-grams (*e.g.*, for all messages), as described *supra*, since full n-grams can be costly and time-consuming to perform for every piece of email.

Using rare character sequences is one way to detect chaff, but it requires making lists of each rare character sequence, of which there can be many. Another way to detect chaff involves detecting high entropy of character sequences in accordance with another aspect of the present invention. Detecting the high entropy of character sequences can be a more cost-effective and efficient manner to identify spam.

Still referring to Fig. 6, the system 600 comprises an entropy detecting component 690 that provides an alternative and/or additional technique to detect chaff. The entropy detecting component can analyze at least a portion of a message via searching for instances of a string of random characters that are indicative of the message being spam.

The entropy of a character sequence is essentially the unlikeliness or the randomness of the sequence. Generally, if the probability P of a character sequence "abc…up to z" is defined as P(abc…z), then the entropy of the sequence is:

$$-\log_2 P(abc\ldots z).$$

The average entropy, or entropy per character (a, b, c,…up to z), which is characterized as:

$$\frac{-\log_2 P(abc\ldots z)}{\text{length}\ (abc\ldots z)}$$

can also be utilized in a similar manner to recognize and identify chaff. The unit of measurement for entropy is "bits."

There are many ways to obtain the probability of a character sequence. For example, a character n-gram language model can be trained on known good email messages, by using a complete corpus of good and bad email, and/or even by using a non-email database. Other heuristics also can be employed to detect high entropy or the average entropy. For instance, lists of common letter pairs or triples (*e.g.*, valid character sequences or 2 and 3 letters, respectively) can be made. Following, the percentage of pairs or triples in any given character sequence that do not occur according to such lists can be included in the entropy determination of that character sequence.

In practice, the relative entropy can also be very useful in providing an advanced and robust spam detection system. More specifically, the average entropy can be detected at the beginning or end of a subject line as being high or relatively high

compared to the middle of the subject line. In practice, for instance, the average entropy at the beginning of a subject line could be 0.5 bits higher than in the middle of the subject line.

Alternatively or in addition, the average entropy at the end or at the beginning of a message can be high compared to the average entropy of the whole message, or can be high compared to the average entropy of the middle of the message. For example, the average entropy at the end of a message could be at least 1 bit higher than the middle of the message (*e.g.*, number of units can be converted into a percentage or factor). Each of these detected events can be a separate feature. Hence, many features are possible.

In addition to random character sequences, a large percentage of spam includes an image instead of text. Images are merely a sequence of 1's and 0's or other numbers. Because of this, spammers can input a minimal amount of static in the image number sequence to pass through conventional spam filtering systems successfully. Thus, the entropy of images can also be determined in a similar manner as the entropy of character sequences. In addition, images detected to be in more than one message can be compared to each other. If they are found to be substantially similar, then all mail including the same or substantially the same image can be blocked.

Finally, the features 670 relating to the entropy events for character sequences and image sequences can be used by the filter training component 680 to train a machine learning filter.

Turning now to Fig. 7, there is a block diagram of an exemplary feature generation system 700 that facilitates spam detection in accordance with an aspect of the present invention. The system 700 comprises a message server 710 whereby a sender 720 sends a message 730, which is delivered to the message server 710 before it reaches its recipient(s). At the message server 710, the message can be parsed by a message header analyzing component 740, an image processing component 750 and/or a message and feature sizing component 760 to yield a myriad of features.

The message header analyzing component 740 analyzes substantially all features of a message header in connection with training a machine learning filter. In particular, machine learning can be employed to automatically identify all useful header features. One approach involves creating features based at least in part upon the presence or

absence of header line types such as "X-Priority", for example, as well as specific header types, such as "X-Priority: 3", for example. In addition, header lines for unsubscribing are useful to identify spam more readily.

Some header lines can be specifically excluded as well according to user preferences. In addition, content of header lines such as the type of mail software being used by the sender can be useful in detecting spam. Examining and analyzing all header lines for their presence, absence, and/or content demonstrates an improvement over traditional machine learning algorithms, which are limited to using features in the subject line and body of messages. Some machine learning algorithms do employ specific features based on email headers but previous approaches have not used all or substantially all possible features in the header lines.

Since spammers like to use images rather than text because it is more difficult and time consuming for filters to analyze images rather than text, the image processing component 750 can be employed to parse out a variety of features based on images included in the message. For example, number of images, location of images (*e.g.*, embedded in the message or externally linked), and/or types of images (*e.g.*, JPGs and/or GIFs) can be ascertained from the message and used as features. In addition, the size (*e.g.*, bytes) as well as X-Y dimensions of the image(s) can be determined with minimal processing, particularly with respect to images embedded in the message.

To avoid blocking legitimate mail containing personal digital photos, special (typically positive) features can be created where the image size matches a common size and/or dimension produced by digital cameras. Features can also relate to the image size in bytes as well as the total area of the images. Finally, features relating to whether the image(s) in the message links to somewhere else (*e.g.*, external to the message) can be indicative of spam since most spammers include external links in their spam.

Alternatively, messages can also include clickable images, whereby the image itself is used as a clickable hyperlink instead of as externally-linked image. In this instance, HTML text in the message contains a tag pattern such as <A HREF="first URL"><IMG SRC="second URL"></A>. It should be appreciated that the first and second URL are different URLs. Hence, features relating to at least a portion of the tag pattern can be used in training a spam filter. In general, HTML attributes and their

respective locations within a tag pattern can be indicative of spam since most spammers try to get around spam filters using images rather than text. Thus, such information can be extracted as features to be used for filter training purposes.

Because very little spam is very big, many different size features can be utilized with the size granularized into one or more buckets by the message sizing component 760. One approach involves features for a message size >100 bytes, >200 bytes, >400 bytes, and up to >***b*** bytes (where ***b*** is an integer greater than or equal to one). This granularization can be based on overlapping buckets such that a message of size 500 bytes would be associated with features for size >100, >200, >400. Alternatively, it can be based on non-overlapping buckets. That is, each bucket pertains to a specific size of a message such that one size feature is associated with each message. For instance, there is a bucket for a message size <100 bytes; 100 = size <200 bytes; 200 = size <400 bytes, up to ***b*** bytes.

In a second approach, the message sizing component can also be applied to subject lines and display names on a smaller size scale since spam and/or spam-like messages tend to have larger subject lines and display names due to the presence of chaff, for example.

Once again, as the message is being parsed and bits of information are being identified, a feature generating component 770 can generate the features from this information and then communicate them or a selected portion of them to be used in conjunction with a filter training component 780.

Various methodologies in accordance with the subject invention will now be described via a series of acts. It is to be understood and appreciated that the present invention is not limited by the order of acts, as some acts may, in accordance with the present invention, occur in different orders and/or concurrently with other acts from that shown and described herein. For example, those skilled in the art will understand and appreciate that a methodology could alternatively be represented as a series of interrelated states or events, such as in a state diagram. Moreover, not all illustrated acts may be required to implement a methodology in accordance with the present invention.

Referring now to Fig. 8, there is illustrated a flow diagram of an exemplary method 800 that facilitates generating features in connection with advanced spam

detection. The method 800 can begin by receiving a message at 810. At least a portion of the message can be parsed at 820 by any suitable email parsing component whereby features corresponding to the message's origination information are generated. The features can be combined into pairs at 830. At 840, the most useful pairs of features can be selected and a filter can be trained with such feature pairs using a machine learning algorithm at 850. The method 800 can be repeated as often as desired to sufficiently train the filter.

There are many features derived from the origination information of a message but some of these features are more useful than others in distinguishing spam from legitimate mail. In particular, features such as the IP address and the related subnet are very difficult for a spammer to modify or disguise. Thus, for legitimate users, these features should match other features such as the sender's alleged machine name and/or the sender's alleged time zone. Accordingly, when these pairs of features are examined, a match among each pair of features indicates a stronger likelihood that the message is legitimate (*e.g.*, not spam). Conversely, when the pair does not match, there is a stronger likelihood that the message is spam.

Turning now to Fig. 9, there is illustrated a flow diagram of an exemplary method 900 of employing a trained filter of Fig. 8 in accordance with an aspect of the present invention. In particular, the method 900 comprises receiving a message at 910, parsing the message to generate one or more origination features at 920, and then pairing up the features at 930 to obtain the most useful feature pairs. At 940, the feature pairs are passed through the machine learning filter to determine whether the particular message is more spam-like. At 950, a verdict can be obtained from the machine learning system as to the spaminess of the message. For example, the verdict can be in the form of a probability corresponding to the likelihood that the message is spam.

Additional features to enhance spam detection can be created such as those depicted in Fig. 10. In Fig. 10, there is illustrated a flow diagram of an exemplary method 1000 that involves receiving one or more messages at 1010, walking through the text of the message and/or subject line to create features for each run of characters up to length ***n*** at 1020 as well as features for each sub-length of character sequences at 1030.

Furthermore, features can be created for character n-grams based on the position of the n-grams (*e.g.*, beginning, end, middle of subject line and/or message body) at 1040. At 1050, features relating to the relative entropy of character sequences as they occur at the end and/or beginning compared to the middle of the subject line and/or message body at can also be generated. High entropy and the entropy per character (*e.g.*, average entropy) of a character sequence can be determined and employed as features as well at 1050. Finally, at 1060, the features can be used to train a machine learning filter. The method 1000 can be repeated until the filter is substantially trained with a desired number of messages, entropy events, and/or character sequences.

Fig. 11 is a flow diagram of an exemplary method 1100 which employs the filter trained in accordance with Fig. 10 to facilitate detecting spam. The method 1100 comprises receiving a message at 1110, scanning at least a portion of the message for character sequences that match and/or do not match a list of valid character sequences (*e.g.*, of multiple lengths) at 1120, and detecting entropy of at least a portion of the character sequences of a message and/or subject line, including those character sequences not found on a list of valid character sequences at 1130. At 1140, the detected events are used as features and passed through a machine learning filter. At 1150, a verdict is obtained from the machine learning system as to whether the message is more spam-like than not.

Turning now to Figs. 12 and 13, there are flow diagrams of exemplary processes 1200 and 1300, respectively, which facilitate generating advanced features for use by a machine learning algorithm in accordance with another aspect of the subject invention. Initially, the method 1200 involves receiving one or more messages at 1210 whereby features can be created by parsing and analyzing the header lines at 1220. In addition, features relating to message and/or feature size (*e.g.*, message size, display name length, subject line) can length) can optionally be created at 1230. At 1240, any images in the message can be analyzed with respect to size, location (internal to message or external link), and/or quantity, among others, to generate further features. Substantially all features created can be employed in the training of a filter using a machine learning system at 1250.

The trained filter of Fig. 12 can be applied to new messages as described in the exemplary process 1300 of Fig. 13. At 1310, one or more messages are received. At 1320, header features are parsed from the message. Optionally, features corresponding to message and/or feature size and/or image characteristics are parsed as well from the message at 1320 and at 1330, respectively. At 1340, these features can be passed through or examined by a machine learning filter. A verdict is obtained at 1350 indicating an amount or a probability of spaminess of the message based at least in part upon the features parsed therefrom.

In order to provide additional context for various aspects of the present invention, Fig. 14 and the following discussion are intended to provide a brief, general description of a suitable operating environment 1410 in which various aspects of the present invention may be implemented. While the invention is described in the general context of computer-executable instructions, such as program modules, executed by one or more computers or other devices, those skilled in the art will recognize that the invention can also be implemented in combination with other program modules and/or as a combination of hardware and software.

Generally, however, program modules include routines, programs, objects, components, data structures, *etc.* that perform particular tasks or implement particular data types. The operating environment 1410 is only one example of a suitable operating environment and is not intended to suggest any limitation as to the scope of use or functionality of the invention. Other well known computer systems, environments, and/or configurations that may be suitable for use with the invention include but are not limited to, personal computers, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, programmable consumer electronics, network PCs, minicomputers, mainframe computers, distributed computing environments that include the above systems or devices, and the like.

With reference to Fig. 14, an exemplary environment 1410 for implementing various aspects of the invention includes a computer 1412. The computer 1412 includes a processing unit 1414, a system memory 1416, and a system bus 1418. The system bus 1418 couples the system components including, but not limited to, the system memory 1416 to the processing unit 1414. The processing unit 1414 can be any of various

available processors. Dual microprocessors and other multiprocessor architectures also can be employed as the processing unit 1414.

The system bus 1418 can be any of several types of bus structure(s) including the memory bus or memory controller, a peripheral bus or external bus, and/or a local bus using any variety of available bus architectures including, but not limited to, 11-bit bus, Industrial Standard Architecture (ISA), Micro-Channel Architecture (MSA), Extended ISA (EISA), Intelligent Drive Electronics (IDE), VESA Local Bus (VLB), Peripheral Component Interconnect (PCI), Universal Serial Bus (USB), Advanced Graphics Port (AGP), Personal Computer Memory Card International Association bus (PCMCIA), and Small Computer Systems Interface (SCSI).

The system memory 1416 includes volatile memory 1420 and nonvolatile memory 1422. The basic input/output system (BIOS), containing the basic routines to transfer information between elements within the computer 1412, such as during start-up, is stored in nonvolatile memory 1422. By way of illustration, and not limitation, nonvolatile memory 1422 can include read only memory (ROM), programmable ROM (PROM), electrically programmable ROM (EPROM), electrically erasable ROM (EEPROM), or flash memory. Volatile memory 1420 includes random access memory (RAM), which acts as external cache memory. By way of illustration and not limitation, RAM is available in many forms such as synchronous RAM (SRAM), dynamic RAM (DRAM), synchronous DRAM (SDRAM), double data rate SDRAM (DDR SDRAM), enhanced SDRAM (ESDRAM), Synchlink DRAM (SLDRAM), and direct Rambus RAM (DRRAM).

Computer 1412 also includes removable/nonremovable, volatile/nonvolatile computer storage media. Fig. 14 illustrates, for example a disk storage 1424. Disk storage 1424 includes, but is not limited to, devices like a magnetic disk drive, floppy disk drive, tape drive, Jaz drive, Zip drive, LS-100 drive, flash memory card, or memory stick. In addition, disk storage 1424 can include storage media separately or in combination with other storage media including, but not limited to, an optical disk drive such as a compact disk ROM device (CD-ROM), CD recordable drive (CD-R Drive), CD rewritable drive (CD-RW Drive) or a digital versatile disk ROM drive (DVD-ROM). To

facilitate connection of the disk storage devices 1424 to the system bus 1418, a removable or non-removable interface is typically used such as interface 1426.

It is to be appreciated that Fig. 14 describes software that acts as an intermediary between users and the basic computer resources described in suitable operating environment 1410. Such software includes an operating system 1428. Operating system 1428, which can be stored on disk storage 1424, acts to control and allocate resources of the computer system 1412. System applications 1430 take advantage of the management of resources by operating system 1428 through program modules 1432 and program data 1434 stored either in system memory 1416 or on disk storage 1424. It is to be appreciated that the present invention can be implemented with various operating systems or combinations of operating systems.

A user enters commands or information into the computer 1412 through input device(s) 1436. Input devices 1436 include, but are not limited to, a pointing device such as a mouse, trackball, stylus, touch pad, keyboard, microphone, joystick, game pad, satellite dish, scanner, TV tuner card, digital camera, digital video camera, web camera, and the like. These and other input devices connect to the processing unit 1414 through the system bus 1418 *via* interface port(s) 1438. Interface port(s) 1438 include, for example, a serial port, a parallel port, a game port, and a universal serial bus (USB). Output device(s) 1440 use some of the same type of ports as input device(s) 1436. Thus, for example, a USB port may be used to provide input to computer 1412, and to output information from computer 1412 to an output device 1440. Output adapter 1442 is provided to illustrate that there are some output devices 1440 like monitors, speakers, and printers among other output devices 1440 that require special adapters. The output adapters 1442 include, by way of illustration and not limitation, video and sound cards that provide a means of connection between the output device 1440 and the system bus 1418. It should be noted that other devices and/or systems of devices provide both input and output capabilities such as remote computer(s) 1444.

Computer 1412 can operate in a networked environment using logical connections to one or more remote computers, such as remote computer(s) 1444. The remote computer(s) 1444 can be a personal computer, a server, a router, a network PC, a workstation, a microprocessor based appliance, a peer device or other common network

node and the like, and typically includes many or all of the elements described relative to computer 1412. For purposes of brevity, only a memory storage device 1446 is illustrated with remote computer(s) 1444. Remote computer(s) 1444 is logically connected to computer 1412 through a network interface 1448 and then physically connected *via* communication connection 1450. Network interface 1448 encompasses communication networks such as local-area networks (LAN) and wide-area networks (WAN). LAN technologies include Fiber Distributed Data Interface (FDDI), Copper Distributed Data Interface (CDDI), Ethernet/IEEE 1102.3, Token Ring/IEEE 1102.5 and the like. WAN technologies include, but are not limited to, point-to-point links, circuit switching networks like Integrated Services Digital Networks (ISDN) and variations thereon, packet switching networks, and Digital Subscriber Lines (DSL).

Communication connection(s) 1450 refers to the hardware/software employed to connect the network interface 1448 to the bus 1418. While communication connection 1450 is shown for illustrative clarity inside computer 1412, it can also be external to computer 1412. The hardware/software necessary for connection to the network interface 1448 includes, for exemplary purposes only, internal and external technologies such as, modems including regular telephone grade modems, cable modems and DSL modems, ISDN adapters, and Ethernet cards.

What has been described above includes examples of the present invention. It is, of course, not possible to describe every conceivable combination of components or methodologies for purposes of describing the present invention, but one of ordinary skill in the art may recognize that many further combinations and permutations of the present invention are possible. Accordingly, the present invention is intended to embrace all such alterations, modifications and variations that fall within the spirit and scope of the appended claims. Furthermore, to the extent that the term "includes" is used in either the detailed description or the claims, such term is intended to be inclusive in a manner similar to the term "comprising" as "comprising" is interpreted when employed as a transitional word in a claim.